Low-Rank Tensor Completion by Approximating the Tensor Average Rank

Zhanliang Wang, Junyu Dong*, Xinguo Liu, Xueying Zeng
Ocean University of China
wangzhanliang@hotmail.com,{dongjunyu;liuxinguo;zxying}@ouc.edu.cn

Abstract

This paper focuses on the problem of low-rank tensor completion, the goal of which is to recover an underlying low-rank tensor from incomplete observations. Our method is motivated by the recently proposed t-product [8] based on any invertible linear transforms. First, we define the new tensor average rank under the invertible real linear transforms. We then propose a tensor completion model using a nonconvex surrogate to approximate the tensor average rank. This surrogate overcomes the discontinuity of the tensor average rank and alleviates the bias problem caused by the convex relaxation. Further, we develop an efficient algorithm to solve the proposed model and establish its convergence. Finally, experimental results on both synthetic and real data demonstrate the superiority of our method.

1. Introduction

Tensors are multi-dimensional arrays, which are the generalization of vectors and matrices to higher dimensions [12]. Due to their high-dimensionality, tensors are used to represent the multi-way data in science and engineering applications. With the advances in data collection and storage capabilities, tensors have received increased attention. They have many applications in computer vision [24], signal processing [22], collaborative filtering [6] and so on.

This paper studies the low-rank tensor completion problem, one of the most concerned problems in tensor applications, aiming to recover a low-rank tensor from incomplete observations. That is because that the elements of tensor data are usually highly correlated in real applications, which indicates that the tensor data are approximately low-rank [12]. Meanwhile, real-world data are often incomplete due to various unavoidable reasons. Low-rank tensor completion is greatly important and has already found many applications in various fields, such as image and video inpainting [15, 31], multitask learning [21], audio classification [19] and many more.

*Junyu Dong is the corresponding author.

Low-rank matrix completion can be considered as a special case of tensor completion. It aims to estimate the missing elements of a partially observed matrix [3]. Mathematically, it can be formulated as

$$\min_{\boldsymbol{X}} \operatorname{rank}(\boldsymbol{X}),\ \text{s.t. } \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{X}) = \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{M}), \tag{1}$$

where $\boldsymbol{M}$ denotes the observed matrix and $\boldsymbol{P}_{\boldsymbol{\Omega}}$ denotes the projection on the observed set $\boldsymbol{\Omega}$. The operation $\operatorname{rank}(\cdot)$ is the rank function of matrix. Due to the discontinuity of the matrix rank function, the matrix rank minimization problem (1) is generally NP-hard and very difficult to solve directly [3]. A commonly used approach is to approximate the matrix rank function by the matrix nuclear norm, which is the tightest convex lower bound of the matrix rank function [20]. Many effective algorithms for solving the obtained convex model are proposed [3, 2, 20]. Matrix completion has been well studied in the recent decade.

It seems natural to extend the existing matrix completion algorithms to the tensor case. Nevertheless, the structure of tensors is more complex than matrices, and the tensor rank cannot be uniformly defined like the matrix case. There are different definitions of tensor rank related to particular tensor decompositions. Based on the CP decomposition, the CP rank is defined as the smallest number of rank-one tensors [9]. However, this tensor rank is in general NP-hard to calculate [4]. Hence, it is very challenging to develop efficient CP rank based tensor completion methods. The traditional approach is to decompose the tensors into matrices and then apply the matrix completion models. Thus the Tucker rank [23], defined on the unfolded matrices, has been studied more widely. A number of tensor completion models based on the Tucker rank are proposed and successfully applied in image processing [15, 27, 7, 28]. However, directly unfolding the high-order tensors into matrices will inevitably destroy the internal structure of tensor data, resulting in information loss and performance degradation [11, 17, 32].

Based on the tensor-tensor product (t-product) and tensor singular value decomposition (t-SVD) [11], the tensor tubal rank has been used to map the intrinsic structure of the tensor data [10, 31, 32, 13]. The tensor nuclear norm

(TNN) is developed and has shown excellent performance in tensor applications [31, 16]. Recently, the authors give a generalized definition of t-product based on any invertible linear transforms [8]. Based on this general t-product, the corresponding definitions of the tensor tubal rank and tensor nuclear norm are proposed in [17]. It has been proved that under suitable assumptions, a three-order tensor can be reconstructed effectively by solving the TNN minimization model [17]. While the TNN based models have achieved great success, they cause a bias problem since the convex relaxation over-penalizes larger singular values in the transform domain [18]. On the other hand, since the tubal rank uses the number of nonzero tubes in transform domain to measure the complexity of tensor, it ignores the sparsity of singular values in transform domain.

To overcome the above drawbacks, we introduce a new tensor rank definition deduced by the invertible real linear transforms, called the tensor average rank. It essentially measures the sparsity of singular values in the transform domain. Due to the discontinuity of the tensor average rank, we develop a new tensor completion model that uses a nonconvex surrogate to approximate the tensor average rank. This surrogate can continuously approximate the tensor average rank while alleviating the bias problem caused by the convex relaxation. A proximal block coordinate descent algorithm is developed to solve our proposed model and some convergence results are established. Experimental results about recovering synthetic data, images, and videos demonstrate that our method is competitive with some existing low-rank tensor completion methods in recovery performance and efficiency.

The rest of this article is organized as follows. Basic notations and preliminaries are provided in Section 2. Section 3 gives the proposed tensor completion model and algorithm. Numerical experiments performed on synthetic data and real-world datasets are shown in Section 4. The summary is drawn in Section 5.

2. Notations and Preliminaries

We introduce our notations and give some necessary preliminary results in this section.

2.1. Notations

In this article, tensors and matrices are denoted by boldface calligraphic letters and boldface uppercase letters, e.g. $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{A}$, respectively. Vectors and scalars are denoted by boldface lowercase letters and lowercase letters, e.g. $\boldsymbol{a}$ and a.

For a three-order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we use $\boldsymbol{\mathcal{A}}_{ijk}$ to represent its (i,j,k)-th entry. The i-th frontal slice of $\boldsymbol{\mathcal{A}}$ is represented by $\boldsymbol{A}^{(i)}$, which is a $n_1 \times n_2$ matrix. The (i,j)-th tube of $\boldsymbol{\mathcal{A}}$ is represented by $\boldsymbol{\mathcal{A}}(i,j,:)$, which is a n_3 sized vector. The inner product of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ with same size is defined as $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{ijk} \boldsymbol{\mathcal{A}}_{ijk} \boldsymbol{\mathcal{B}}_{ijk}$. Similar to matrix algebra, two commonly used norms of tensors can be defined. We denote the Frobenius norm induced by above tensor product by $\|\boldsymbol{\mathcal{A}}\|_{\mathrm{F}} = \sqrt{\sum_{ijk} \boldsymbol{\mathcal{A}}_{ijk}^2}$, and the infinity norm by $\|\boldsymbol{\mathcal{A}}\|_\infty = \max_{ijk} |\boldsymbol{\mathcal{A}}_{ijk}|$. We transform a three-order tensor to a block diagonal matrix by operator $\mathtt{bdiag}$. Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, then $\mathtt{bdiag}(\boldsymbol{\mathcal{A}})$ is an $n_1 n_3 \times n_2 n_3$ matrix defined as

$$\mathtt{bdiag}(\boldsymbol{\mathcal{A}}) = \begin{bmatrix} \boldsymbol{A}^{(1)} & & & \\ & \boldsymbol{A}^{(2)} & & \\ & & \ddots & \\ & & & \boldsymbol{A}^{(n_3)} \end{bmatrix}.$$

2.2. Preliminary Results

Definition 2.1 (Mode-3 product) *[14] The mode-3 (matrix) product of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\boldsymbol{U} \in \mathbb{R}^{n_3 \times n_3}$ is given by*

$$(\boldsymbol{\mathcal{A}} \times_3 \boldsymbol{U})_{ijk} = \sum_{s=1}^{n_3} \boldsymbol{\mathcal{A}}_{ijs} \boldsymbol{U}_{ks}.$$

Definition 2.2 (Face-wise product) *[8] The face-wise product of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times l \times n_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{l \times n_2 \times n_3}$, denoted by $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \odot \boldsymbol{\mathcal{B}}$, is given by*

$$\boldsymbol{C}^{(i)} = \boldsymbol{A}^{(i)} \boldsymbol{B}^{(i)},\ i = 1, \cdots, n_3.$$

The work [8] introduces the definition of t-product based on any invertible linear transform. In this work, we consider the linear transform $\mathcal{L} : \mathbb{R}^{n_1 \times n_2 \times n_3} \to \mathbb{R}^{n_1 \times n_2 \times n_3}$, which is defined as

$$\bar{\boldsymbol{\mathcal{A}}} = \mathcal{L}(\boldsymbol{\mathcal{A}}) = \boldsymbol{\mathcal{A}} \times_3 \boldsymbol{L}, \tag{2}$$

where $\boldsymbol{L} \in \mathbb{R}^{n_3 \times n_3}$ can be any invertible matrix. Clearly, $\mathcal{L}$ is invertible with the inverse transform defined as

$$\boldsymbol{\mathcal{A}} = \mathcal{L}^{-1}(\bar{\boldsymbol{\mathcal{A}}}) = \bar{\boldsymbol{\mathcal{A}}} \times_3 \boldsymbol{L}^{-1}.$$

We will denote by $\bar{\boldsymbol{A}}$ the block diagonal matrix of $\bar{\boldsymbol{\mathcal{A}}}$, i.e., $\bar{\boldsymbol{A}} = \mathtt{bdiag}(\bar{\boldsymbol{\mathcal{A}}})$. With the face-wise product and invertible linear transform $\mathcal{L}$, the definition of t-product can be given as follows.

Definition 2.3 (T-product) *[8, 17] Suppose $\mathcal{L}$ is an invertible linear transform in* (2)*, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times l \times n_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{l \times n_2 \times n_3}$. Then the t-product of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ under $\mathcal{L}$, denoted by $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} *_{\mathcal{L}} \boldsymbol{\mathcal{B}}$, is defined such that $\mathcal{L}(\boldsymbol{\mathcal{C}}) = \mathcal{L}(\boldsymbol{\mathcal{A}}) \odot \mathcal{L}(\boldsymbol{\mathcal{B}})$.*

Because $\mathcal{L}(\boldsymbol{\mathcal{C}}) = \mathcal{L}(\boldsymbol{\mathcal{A}}) \odot \mathcal{L}(\boldsymbol{\mathcal{B}})$, we can get that $\bar{\boldsymbol{C}} = \bar{\boldsymbol{A}} \bar{\boldsymbol{B}}$. The tensor-tensor product in tensor domain is essentially a matrix-matrix product under linear transform. It can be efficiently computed by Algorithm 1.

We need some further definitions, including tensor transpose, identity tensor, orthogonal tensor, to define the t-SVD. They are the high-order extensions of the definitions in matrix theory.

Algorithm 1 General T-product [8, 17]

Input: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times l \times n_3}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{l \times n_2 \times n_3}$ and linear transform $\mathcal{L}$ in (2).

1. Calculate $\bar{\boldsymbol{\mathcal{A}}} = \mathcal{L}(\boldsymbol{\mathcal{A}})$ and $\bar{\boldsymbol{\mathcal{B}}} = \mathcal{L}(\boldsymbol{\mathcal{B}})$.
2. Calculate all slices $\bar{\boldsymbol{C}}^{(i)}$ by
for $i = 1, \cdots, n_3$ **do**
$\bar{\boldsymbol{C}}^{(i)} = \bar{\boldsymbol{A}}^{(i)} \bar{\boldsymbol{B}}^{(i)}$;
end for
3. Calculate $\boldsymbol{\mathcal{C}} = \mathcal{L}^{-1}(\bar{\boldsymbol{\mathcal{C}}})$.

Output: $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

Definition 2.4 (Tensor transpose) *[8, 17] Suppose $\mathcal{L}$ is an invertible linear transform in* (2)*, the transpose of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ under $\mathcal{L}$ is a $n_2 \times n_1 \times n_3$ tensor $\boldsymbol{\mathcal{A}}^\top$ which satisfies $\mathcal{L}(\boldsymbol{\mathcal{A}}^\top)^{(i)} = (\mathcal{L}(\boldsymbol{\mathcal{A}})^{(i)})^\top$, $i = 1, \cdots, n_3$.*

Definition 2.5 (Identity tensor) *[8, 17] Suppose $\mathcal{L}$ is an invertible linear transform in* (2)*, the tensor $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{n \times n \times n_3}$ is an identity tensor under $\mathcal{L}$ if all frontal slices of $\mathcal{L}(\boldsymbol{\mathcal{I}})$ are identity matrices.*

Definition 2.6 (Orthogonal tensor) *[8, 17] Suppose $\mathcal{L}$ is an invertible linear transform in* (2)*, the orthogonal tensor $\boldsymbol{\mathcal{O}}$ under $\mathcal{L}$ is a $n \times n \times n_3$ tensor which satisfies $\boldsymbol{\mathcal{O}} *_{\mathcal{L}} \boldsymbol{\mathcal{O}}^\top = \boldsymbol{\mathcal{O}}^\top *_{\mathcal{L}} \boldsymbol{\mathcal{O}} = \boldsymbol{\mathcal{I}}$.*

Definition 2.7 (F-diagonal) *[11] A three-order tensor is a f-diagonal tensor if its all frontal slices are diagonal matrices.*

Theorem 2.1 (T-SVD) *[8, 17] Suppose $\mathcal{L}$ is an invertible linear transform in* (2)*, and $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. There exist tensors $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$, $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, and $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ such that*

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{U}} *_{\mathcal{L}} \boldsymbol{\mathcal{S}} *_{\mathcal{L}} \boldsymbol{\mathcal{V}}^\top, \tag{3}$$

where $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ are orthogonal tensors under $\mathcal{L}$, and $\boldsymbol{\mathcal{S}}$ is a f-diagonal tensor.

Note that $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{U}} *_{\mathcal{L}} \boldsymbol{\mathcal{S}} *_{\mathcal{L}} \boldsymbol{\mathcal{V}}^\top$ in tensor domain is actually equivalent to $\bar{\boldsymbol{A}} = \bar{\boldsymbol{U}} \bar{\boldsymbol{S}} \bar{\boldsymbol{V}}^\top$ in transform domain. That means that we can obtain the t-SVD of a tensor by calculating the matrix SVD. Technically, we need to calculate the matrix SVD of all frontal slice $\bar{\boldsymbol{A}}^{(i)}$ in transform domain, i.e., $\bar{\boldsymbol{A}}^{(i)} = \bar{\boldsymbol{U}}^{(i)} \bar{\boldsymbol{S}}^{(i)} (\bar{\boldsymbol{V}}^{(i)})^\top$, then transform $\bar{\boldsymbol{\mathcal{U}}}$, $\bar{\boldsymbol{\mathcal{S}}}$, $\bar{\boldsymbol{\mathcal{V}}}$ to tensor domain by the inverse transform $\mathcal{L}^{-1}$. See Algorithm 2 for the details of calculating the t-SVD.

For any invertible linear transform in (2), it holds that $\mathcal{L}(0) = \mathcal{L}^{-1}(0) = 0$. Then $\boldsymbol{\mathcal{S}}$ and $\bar{\boldsymbol{\mathcal{S}}}$ are f-diagonal, the tensor tubal rank is defined as the number of nonzero singular tubes.

Algorithm 2 General T-SVD [8, 17]

Input: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and linear transform $\mathcal{L}$ in (2).

1. Calculate $\bar{\boldsymbol{\mathcal{A}}} = \mathcal{L}(\boldsymbol{\mathcal{A}})$.
2. Calculate all slices $\bar{\boldsymbol{U}}^{(i)}$, $\bar{\boldsymbol{S}}^{(i)}$, $\bar{\boldsymbol{V}}^{(i)}$ by
for $i = 1, \cdots, n_3$ **do**
$[\bar{\boldsymbol{U}}^{(i)}, \bar{\boldsymbol{S}}^{(i)}, \bar{\boldsymbol{V}}^{(i)}] = \text{SVD}(\bar{\boldsymbol{A}}^{(i)})$;
end for
3. Calculate $\boldsymbol{\mathcal{U}} = \mathcal{L}^{-1}(\bar{\boldsymbol{\mathcal{U}}})$, $\boldsymbol{\mathcal{S}} = \mathcal{L}^{-1}(\bar{\boldsymbol{\mathcal{S}}})$, and $\boldsymbol{\mathcal{V}} = \mathcal{L}^{-1}(\bar{\boldsymbol{\mathcal{V}}})$.

Output: $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$, $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, and $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$.

Definition 2.8 (Tensor tubal rank) *[17] Suppose $\mathcal{L}$ is an invertible linear transform in* (2)*, and $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. The tensor tubal rank of $\boldsymbol{\mathcal{A}}$ under $\mathcal{L}$ is defined as the number of nonzero singular tubes of $\boldsymbol{\mathcal{S}}$, i.e.,*

$$rank_t(\boldsymbol{\mathcal{A}}) = \sharp\{i | \boldsymbol{\mathcal{S}}(i, i, :) \neq \boldsymbol{0}\}, \tag{4}$$

where $\boldsymbol{\mathcal{S}}$ is from the t-SVD of $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{U}} *_{\mathcal{L}} \boldsymbol{\mathcal{S}} *_{\mathcal{L}} \boldsymbol{\mathcal{V}}^T$.

Based on the t-SVD, the tensor tubal rank gives a scalar measure of tensor complexity. Further, the singular values of all frontal slices in the transform domain can be obtained by calculating the t-SVD. We then define a new tensor rank to characterize the sparsity of the singular values in the transform domain.

Definition 2.9 (Tensor average rank) *Suppose $\mathcal{L}$ is an invertible linear transform in* (2)*, and $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. The tensor average rank of $\boldsymbol{\mathcal{A}}$ under $\mathcal{L}$ is defined as the average across the ranks of all frontal slices in the transform domain, i.e.,*

$$rank_a(\boldsymbol{\mathcal{A}}) = \frac{1}{n_3} \sum_{i=1}^{n_3} rank(\bar{\boldsymbol{A}}^{(i)}). \tag{5}$$

The tensor average rank measures the sparsity of the singular values in the transform domain. Let $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{U}} *_{\mathcal{L}} \boldsymbol{\mathcal{S}} *_{\mathcal{L}} \boldsymbol{\mathcal{V}}^{\mathrm{T}}$ be the t-SVD of $\boldsymbol{\mathcal{A}}$ under $\mathcal{L}$, $\text{rank}_{\text{a}}(\boldsymbol{\mathcal{A}})$ is equal to the number of nonzero elements of $\bar{\boldsymbol{\mathcal{S}}}$ times the factor $\frac{1}{n_3}$, i.e.,

$$\text{rank}_{\text{a}}(\boldsymbol{\mathcal{A}}) = \frac{1}{n_3} \sum_{i,k} \delta(\bar{\boldsymbol{\mathcal{S}}}_{iik}), \tag{6}$$

where $\delta(x)$ is the Kronecker delta function defined as

$$\delta(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x = 0. \end{cases} \tag{7}$$

In next section, we study the tensor completion problem by an approximate tensor average rank minimization model.

For this purpose, we set an assumption on the linear transform in (2), e.g.,

$$\boldsymbol{L}^\top \boldsymbol{L} = \boldsymbol{L}\boldsymbol{L}^\top = \boldsymbol{I}_{n_3}, \tag{8}$$

where $\boldsymbol{I}_{n_3}$ is the $n_3 \times n_3$ sized identity matrix. If the transform $\mathcal{L}$ in (2) satisfies (8), we call it real orthogonal transform. Under this transform, we can get that

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \langle \bar{\boldsymbol{\mathcal{A}}}, \bar{\boldsymbol{\mathcal{B}}} \rangle = \langle \bar{\boldsymbol{A}}, \bar{\boldsymbol{B}} \rangle, \tag{9}$$

and

$$\|\boldsymbol{\mathcal{A}}\|_{\mathrm{F}} = \|\bar{\boldsymbol{\mathcal{A}}}\|_{\mathrm{F}} = \|\bar{\boldsymbol{A}}\|_{\mathrm{F}}. \tag{10}$$

3. Main Result

In this section, we introduce a new nonconvex relaxation model for tensor completion. We then develop an efficient algorithm to solve the proposed model and establish its convergence.

The definition of tensor averaged rank in (5) has a factor $\frac{1}{n_3}$. Note that this factor has no effect on minimizing the tensor average rank. For brevity, we temporarily omit this factor in this section and modify the tensor average rank as

$$\mathrm{rank_a}(\boldsymbol{\mathcal{A}}) = \sum_{i=1}^{n_3} \mathrm{rank}(\bar{\boldsymbol{A}}^{(i)}) = \sum_{i,k} \delta(\bar{\boldsymbol{\mathcal{S}}}_{iik}). \tag{11}$$

3.1. The Proposed Model

Using the tensor average rank to measure the sparsity of singular values, we solve the following tensor completion model

$$\min_{\boldsymbol{\mathcal{X}}} \mathrm{rank_a}(\boldsymbol{\mathcal{X}}),\ \text{s.t. } \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{X}}) = \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{M}}), \tag{12}$$

where $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ denotes the observed tensor data and $\boldsymbol{P}_{\boldsymbol{\Omega}}$ denotes the projection on the observed set $\boldsymbol{\Omega}$. Let $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{U}} *_{\mathcal{L}} \boldsymbol{\mathcal{S}} *_{\mathcal{L}} \boldsymbol{\mathcal{V}}^{\mathrm{T}}$ be the t-SVD of $\boldsymbol{\mathcal{X}}$, we can equivalently reformulate (12) in transform domain as

$$\min_{\boldsymbol{\mathcal{X}}} \sum_{i,k} \delta(\bar{\boldsymbol{\mathcal{S}}}_{iik}),\ \text{s.t. } \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{X}}) = \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{M}}). \tag{13}$$

Directly solving (13) is difficult due to the discontinuous nature of the Kronecker delta function. As an extension of the low-rank matrix completion, the work [17] presents the tensor nuclear norm minimization model, which uses the ℓ_1 norm as a surrogate of $\delta(x)$ in (13). In this paper, we use a nonconvex function to approximate the Kronecker delta function and propose a novel surrogate of the tensor average rank. For $\lambda > 0$, we let

$$\phi_\lambda(x) := \min\{1, \frac{1}{2\lambda}x^2\},\ x \geq 0. \tag{14}$$

Compared with the popular nonconvex regularizers in [26], the function $\phi_\lambda(x)$ relaxes $\delta(x)$ near zero using l_2-norm. Hence, it is not concave. However, $\phi_\lambda(x)$ overcomes the discontinuity of $\delta(x)$ near zero, while still keeps unbiased when x is greater than the threshold $\sqrt{2\lambda}$. Moreover, for any $x \geq 0$, $\phi_\lambda(x)$ closely matches $\delta(x)$ as λ decreasing to zero, i.e., $\lim_{\lambda \to 0^+} \phi_\lambda(x) = \delta(x)$. Replacing the Kronecker delta function with $\phi_\lambda(x)$ in (13), the tensor average rank of $\boldsymbol{\mathcal{X}}$ can be approximated by

$$\Phi_\lambda(\boldsymbol{\mathcal{X}}) = \sum_{i,k} \phi_\lambda(\bar{\boldsymbol{\mathcal{S}}}_{iik}). \tag{15}$$

Using $\Phi_\lambda(\boldsymbol{\mathcal{X}})$ as a surrogate of the tensor average rank, we propose the following low-rank tensor completion model

$$\min_{\boldsymbol{\mathcal{X}}} \Phi_\lambda(\boldsymbol{\mathcal{X}}),\ \text{s.t. } \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{X}}) = \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{M}}). \tag{16}$$

3.2. Tensor Hard Thresholding

We will rewrite the proposed model (16) into an equivalent form that facilitates us to develop efficient algorithms. To this aim, we introduce the hard thresholding rule in the tensor case. Let $\lambda > 0$ and x be a scalar variable, the hard thresholding operator $\mathcal{H}_\lambda$ is defined as

$$\mathcal{H}_\lambda(x) = \begin{cases} x & \text{if } |x| > \sqrt{2\lambda}, \\ 0 & \text{if } |x| \leq \sqrt{2\lambda}. \end{cases}$$

It can be extended to tensor space by element-wise operation, i.e.,

$$(\boldsymbol{\mathcal{H}}_\lambda(\boldsymbol{\mathcal{A}}))_{ijk} = \mathcal{H}_\lambda(\boldsymbol{\mathcal{A}}_{ijk}). \tag{17}$$

Let $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{U}} *_{\mathcal{L}} \boldsymbol{\mathcal{S}} *_{\mathcal{L}} \boldsymbol{\mathcal{V}}^{\mathrm{T}}$ be the t-SVD of $\boldsymbol{\mathcal{X}}$. For any $\lambda > 0$, we define the tensor hard thresholding (THT) operator $\boldsymbol{\mathcal{D}}_\lambda : \mathbb{R}^{n_1 \times n_2 \times n_3} \to \mathbb{R}^{n_1 \times n_2 \times n_3}$ as

$$\boldsymbol{\mathcal{D}}_\lambda(\boldsymbol{\mathcal{X}}) = \boldsymbol{\mathcal{U}} *_{\mathcal{L}} \boldsymbol{\mathcal{S}}_\lambda *_{\mathcal{L}} \boldsymbol{\mathcal{V}}^\top, \tag{18}$$

where

$$\boldsymbol{\mathcal{S}}_\lambda = \mathcal{L}^{-1}(\boldsymbol{\mathcal{H}}_\lambda(\mathcal{L}(\boldsymbol{\mathcal{S}}))), \tag{19}$$

with $\boldsymbol{\mathcal{H}}_\lambda$ being defined in (17).

The THT operator applies the hard-thresholding rule to the singular values of each frontal slice of $\mathcal{L}(\boldsymbol{\mathcal{S}})$, which can shrink the small singular values to zero. The following theorem demonstrates that the THT operator is essentially the proximal operator related to the tensor average rank.

Theorem 3.1 *Suppose $\mathcal{L}$ is a real orthogonal transform in* (2) *and $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. For any $\lambda > 0$, the THT operator obeys*

$$\boldsymbol{\mathcal{D}}_\lambda(\boldsymbol{\mathcal{X}}) \in \arg\min_{\boldsymbol{\mathcal{Y}}} \frac{1}{2\lambda}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \mathit{rank_a}(\boldsymbol{\mathcal{Y}}). \tag{20}$$

Theorem 3.1 indicates that the THT operator $\boldsymbol{\mathcal{D}}_\lambda(\boldsymbol{\mathcal{X}})$ has its closed-form. The tensor operator is actually an extension

Algorithm 3 Tensor Hard Thresholding (THT)

Input: $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, real orthogonal transform $\mathcal{L}$, and $\lambda > 0$.

1. Calculate $\bar{\boldsymbol{\mathcal{X}}} = \mathcal{L}(\boldsymbol{\mathcal{X}})$.

2. Apply matrix hard thresholding to all slices $\bar{\boldsymbol{X}}^{(i)}$ by

for $i = 1, \cdots, n_3$ **do**

$[\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}] = \text{SVD}(\bar{\boldsymbol{X}}^{(i)})$;

$\bar{\boldsymbol{D}}^{(i)} = \boldsymbol{U}\boldsymbol{H}_\lambda(\boldsymbol{S})\boldsymbol{V}^\top$;

end for

3. Calculate $\boldsymbol{\mathcal{D}}_\lambda(\boldsymbol{\mathcal{X}}) = \mathcal{L}^{-1}(\bar{\boldsymbol{\mathcal{D}}})$.

Output: $\boldsymbol{\mathcal{D}}_\lambda(\boldsymbol{\mathcal{X}}) \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

of matrix hard thresholding [5]. We summarize the details of computing $\boldsymbol{\mathcal{D}}_\lambda(\boldsymbol{\mathcal{X}})$ in Algorithm 3.

The next proposition gives a separable structure of Φ_λ. Moreover, Φ_λ can accurately approximate the tensor average rank when the parameter λ is small to a certain extent.

Proposition 3.1 *Suppose $\mathcal{L}$ is a real orthogonal transform in* (2) *and $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. For any $\lambda > 0$, we have*

$$\Phi_\lambda(\boldsymbol{\mathcal{X}}) = \min_{\boldsymbol{\mathcal{Y}}} \frac{1}{2\lambda}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \mathit{rank}_a(\boldsymbol{\mathcal{Y}}). \tag{21}$$

Moreover, $\Phi_\lambda(\boldsymbol{\mathcal{X}}) = \mathit{rank}_a(\boldsymbol{\mathcal{X}})$ when $\lambda \leq \frac{\min_{i,k} \bar{\boldsymbol{\mathcal{S}}}_{iik}^2}{2}$.

Proposition 3.1 ensures that the proposed model (16) can be converted as the following two-variables problem

$$\min_{\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{X}}} \{\frac{1}{2\lambda}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_{\text{F}}^2 + \text{rank}_{\text{a}}(\boldsymbol{\mathcal{Y}})\}, \text{ s.t. } \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{X}}) = \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{M}}). \tag{22}$$

Suppose that $(\boldsymbol{\mathcal{Y}}_*, \boldsymbol{\mathcal{X}}_*)$ is a solution of (22), by Proposition 3.1, it must hold that $\boldsymbol{\mathcal{X}}_*$ solves (16) and $\boldsymbol{\mathcal{Y}}_* = \boldsymbol{\mathcal{D}}_\lambda(\boldsymbol{\mathcal{X}}_*)$. Thus we can solve (22) instead of solving (16) directly. The obvious advantage of (22) is that its objective function has a good separable structure which helps us to use the alternating minimization type algorithms to solve it.

3.3. Algorithm

Problem (22) is a joint minimization problem with respect to variable $(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{X}})$ and it can be solved by the block coordinate descent (BCD) algorithm. Note that the convergence of direct BCD algorithm requires various conditions of the objective function [30, 25]. That motivates us to propose a proximal BCD algorithm to solve (22). Specifically, at each iteration we generate $\tilde{\boldsymbol{\mathcal{X}}}_k := (1-\mu)\boldsymbol{\mathcal{Y}}_k + \mu\boldsymbol{\mathcal{X}}_k$ with $\mu \in (0, 1)$ being a weight which balances both terms $\boldsymbol{\mathcal{Y}}_k$ and $\boldsymbol{\mathcal{X}}_k$, the iterative scheme of the proposed algorithm for (22) is indicated as below

$$\begin{cases} \boldsymbol{\mathcal{Y}}_{k+1} \in \arg\min_{\boldsymbol{\mathcal{Y}}} \frac{1}{2\lambda}\|\boldsymbol{\mathcal{Y}} - \tilde{\boldsymbol{\mathcal{X}}}_k\|_{\text{F}}^2 + \mu\text{rank}_{\text{a}}(\boldsymbol{\mathcal{Y}}), \\ \boldsymbol{\mathcal{X}}_{k+1} = \arg\min_{\boldsymbol{\mathcal{X}}} \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}_{k+1}\|_{\text{F}}^2, \text{s.t.} \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{X}}) = \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{M}}). \end{cases} \tag{23}$$

We next show that both subproblems in (23) have closed-form solutions.

The first subproblem of updating $\boldsymbol{\mathcal{Y}}_{k+1}$, by Theorem 3.1, can be directly solved by performing the THT operator on $\tilde{\boldsymbol{\mathcal{X}}}_k$, e.g.,

$$\boldsymbol{\mathcal{Y}}_{k+1} = \boldsymbol{\mathcal{D}}_{\lambda\mu}(\tilde{\boldsymbol{\mathcal{X}}}_k). \tag{24}$$

Therefore, $\boldsymbol{\mathcal{Y}}_{k+1}$ can be efficiently computed by Algorithm 3. The second subproblem of updating $\boldsymbol{\mathcal{X}}_{k+1}$ can be exactly solved by

$$\boldsymbol{\mathcal{X}}_{k+1} = \boldsymbol{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{M}}) + \boldsymbol{P}_{\boldsymbol{\Omega}^{\text{c}}}(\boldsymbol{\mathcal{Y}}_{k+1}), \tag{25}$$

where $\boldsymbol{\Omega}^{\text{c}}$ is the complement of $\boldsymbol{\Omega}$.

3.4. Convergence Analysis

Let $E(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{X}})$ denote the objective function of (22), i.e.,

$$E(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{X}}) = \frac{1}{2\lambda}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_{\text{F}}^2 + \text{rank}_{\text{a}}(\boldsymbol{\mathcal{Y}}). \tag{26}$$

The following theorem demonstrates some crucial properties of the sequences $\{E(\boldsymbol{\mathcal{Y}}_k, \boldsymbol{\mathcal{X}}_k)\}$ and $\{(\boldsymbol{\mathcal{Y}}_k, \boldsymbol{\mathcal{X}}_k)\}$ generated by (23).

Theorem 3.2 *Let $\{(\boldsymbol{\mathcal{Y}}_k, \boldsymbol{\mathcal{X}}_k)\}$ be generated by* (23)*, then it satisfies the following properties:*

(1) $\{E(\boldsymbol{\mathcal{Y}}_k, \boldsymbol{\mathcal{X}}_k)\}$ is monotonically decreasing and therefore converges.

(2) $\lim_{k\to\infty} \|\boldsymbol{\mathcal{Y}}_{k+1} - \boldsymbol{\mathcal{Y}}_k\|_F = 0$, $\lim_{k\to\infty} \|\boldsymbol{\mathcal{X}}_{k+1} - \boldsymbol{\mathcal{X}}_k\|_F = 0$.

(3) The tensor average rank of $\boldsymbol{\mathcal{Y}}_k$ keeps invariant after a finite iteration, i.e., there exists two positive constants K and r such that $\mathit{rank}_a(\boldsymbol{\mathcal{Y}}_k) = r$ for all $k > K$.

Theorem 3.2 shows that the average rank of the sequence $\boldsymbol{\mathcal{Y}}^k$ stabilizes after finitely many iterations. This is because the location of nonzero singular values in the transform domain remains unchanged after limited iterations. It implies that the tubal rank also stabilizes and converges.

3.5. Algorithm Implementation

The parameter λ is crucial for the proposed model (16) and algorithm (23). It reflects the nonconvexity of $\Phi_\lambda(\boldsymbol{\mathcal{X}})$ and measures its approximation accuracy to $\text{rank}_{\text{a}}(\boldsymbol{\mathcal{X}})$. By Proposition 3.1, the function $\Phi_\lambda(\boldsymbol{\mathcal{X}})$ is equivalent to the tensor average rank when λ is small enough. Unfortunately,

the nonconvexity of model (16) becomes stronger as λ decreases. Therefore, directly choosing a small λ in algorithm (23) would make the algorithm converge slowly and trap in a shallow local minima.

To avoid the above issue, a coarse-to-fine continuation minimization strategy is used in our algorithm. It has been usually applied to solve some nonconvex problems, which can empirically yield a better local minimal solution [1]. The details of algorithm implementation are summarized in Algorithm 4.

Algorithm 4 Low-rank tensor completion

Input: Observed tensor $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, and parameters μ, $\lambda_{\min}$, $\lambda_{\max}$, ρ, δ, ε.
Initialize: $\mathcal{Y}_0$, $\mathcal{X}_0$, $\lambda_0 = \lambda_{\max}$.
while not converged **do**
1. Update $\mathcal{Y}_{k+1}$ by (24).
2. Update $\mathcal{X}_{k+1}$ by (25).
3. Update λ_{k+1} by $\lambda_{k+1} = \max(\rho\lambda_k, \lambda_{\min})$ if $\min(\|\mathcal{Y}_{k+1} - \mathcal{Y}_k\|_\infty, \|\mathcal{X}_{k+1} - \mathcal{X}_k\|_\infty) \leq \delta$.
4. Check the stopping criteria: $\min(\|\mathcal{Y}_{k+1} - \mathcal{Y}_k\|_\infty, \|\mathcal{X}_{k+1} - \mathcal{X}_k\|_\infty) \leq \varepsilon$, $\min(\|\mathcal{Y}_{k+1} - \mathcal{Y}_k\|_\infty, \|\mathcal{X}_{k+1} - \mathcal{X}_k\|_\infty) \neq 0$.

end while
Output: $\mathcal{Y}_{k+1}$, $\mathcal{X}_{k+1}$.

As shown in Algorithm 4, we start our algorithm with a relatively large $\lambda_{\max}$ and then progressively decrease it when the algorithm achieves a certain level of convergence. Then the solution at the current iteration is used as a feasible initial pair to minimize $\Phi_\lambda(\mathcal{X})$ with a smaller λ at the next iteration. Moreover, the second part of the stopping criteria ensures that the parameter λ can be reduced to a small value $\lambda_{\min}$ effectively. Finally, our algorithm will converge to a low-rank solution since the $\lambda_{\min}$ is sufficiently small.

Here we give a simple example to verify the effect of the coarse-to-fine continuation minimization idea. We generate a tensor $\mathcal{X} \in \mathbb{R}^{10\times10\times10}$ with $\text{rank}_a(\mathcal{X}) = 1$ and the sampling rate $p = 0.5$. Figure 1 shows the relative square errors of the two experiments: (1) fixed-parameter $\lambda = \lambda_{\min} = 1$; (2) Algorithm 4. We can observe that Algorithm 4 achieves excellent recovery performance while setting a fixed small λ fails to get a good solution. Furthermore, Algorithm 4 converges much faster than the algorithm with a fixed small λ. As a result, the idea of gradually updating λ can accelerate the algorithm (23).

4. Experiments

In this section, we conduct some numerical experiments to demonstrate the advantages of our proposed model. We first explore the recovery capability of the proposed model. Then we use it to recover real data and compare the performance with some popular tensor completion methods. With regard to the selection of transforms, we use the discrete cosine transform (DCT) in our experiments.

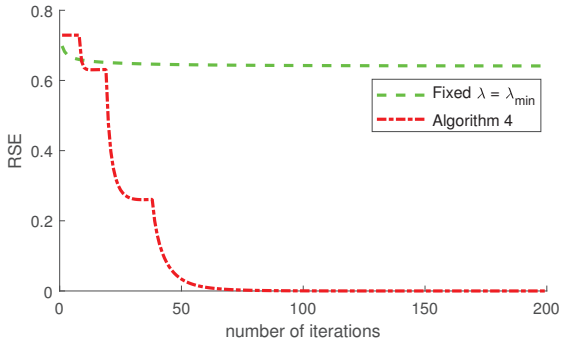

Figure 1. The effect of adaptively updating parameter λ.

4.1. Synthetic Experiments

We investigate the capability of our model by recovering synthetic data. Here, we generate an average rank r tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times n_3}$ by $\mathcal{X} = \mathcal{P} *_{\mathcal{L}} \mathcal{Q}$, where the elements of $\mathcal{P} \in \mathbb{R}^{n_1\times r\times n_3}$ and $\mathcal{Q} \in \mathbb{R}^{r\times n_2\times n_3}$ are obtained from a normal distribution. In this way, the average rank is made equal to the tubal rank, so that we can compare our model with the TNN-DCT model [17]. Finally, we obtain the observed tensor by taking $pn_1n_2n_3$ elements uniformly from $\mathcal{X}$, where p is the sampling rate. The Relative Square Error (RSE) is defined as

$$\text{RSE} = \frac{\|\mathcal{X} - \hat{\mathcal{X}}\|_F}{\|\mathcal{X}\|_F},$$

where $\hat{\mathcal{X}}$ denotes the solution obtained by an algorithm. If $\text{RSE} \leq 10^{-3}$, then we claim that the recovery is successful.

We examine the recovery phenomenon by varying the average rank of $\mathcal{X}$ and the sampling rate p. In this experiment, we consider tensor $\mathcal{X} \in \mathbb{R}^{n\times n\times n}$ of two different sizes: (1) $n = 40$ and (2) $n = 50$. We choose $r = [1, 2, \cdots, 32]$ in the first case and $r = [1, 2, \cdots, 38]$ in the second case. We set $p = [0.01 : 0.01 : 0.99]$ in both cases. For each (r, p) pair, we repeat the experiments for 10 times.

The exhaustive comparison between TNN-DCT and our method is shown in Figure 2. The color of each cell reflects the empirical recovery rate ranging from 0 to 1 (blue $= 0$; yellow $= 1$). It can be seen that the yellow region of our method is more extensive than that of TNN-DCT in both cases. For example, when the sampling rate is 0.5, the maximum rank which can be recovered by our method is four higher than that of TNN-DCT in both cases. These results verify the effectiveness of our proposed model.

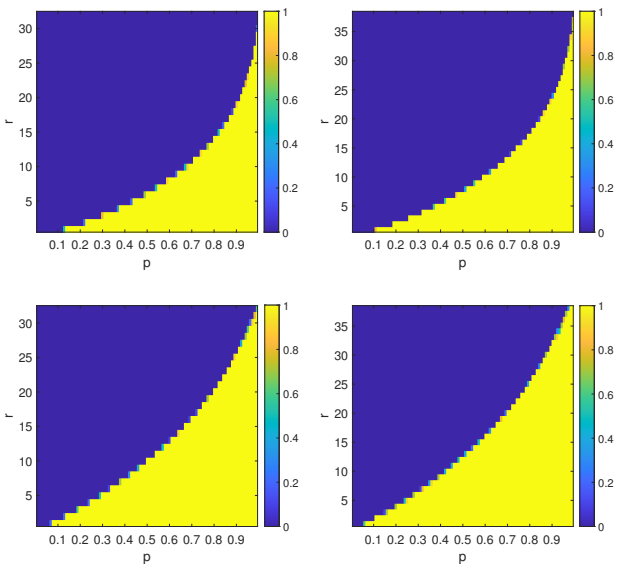

Figure 2. Comparison our model with TNN-DCT. Upper left: The TNN-DCT model in the case $n = 40$; Upper right: The TNN-DCT model in the case $n = 50$; Lower left: Our model in the case $n = 40$; Lower right: Our model in the case $n = 50$.

4.2. Applications to Real Data

This method can be applied in color image, hyperspectral image and video inpainting tasks, and background initialization in videos. In this subsection, we use our tensor completion method to recover incomplete color images and videos. We test five low-rank tensor completion methods and compare their performance: (1) HaLRTC [15]: high accuracy Tucker rank based method; (2) SPC [29]: smooth PARAFAC tensor completion method; (3) T-SVD [31]: t-SVD based method; (4) TNN [16]: TNN based method (via the discrete Fourier transform); (5) TNN-DCT [17]: TNN based method (via the discrete cosine transform); (6) Ours: our proposed tensor completion method. We apply the commonly used PSNR metric, defined as

$$\text{PSNR} = 10\log_{10}\left(\frac{\|\boldsymbol{\mathcal{X}}\|_{\infty}^2}{\frac{1}{n_1 n_2 n_3}\|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\|_{\text{F}}^2}\right),$$

to evaluate the recovery performance.

4.2.1 Image Inpainting

We apply our tensor completion model to the image inpainting task. For a color image of size $m \times n$, it can be regarded as a three-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{m\times 3\times n}$, where the lateral slice corresponds to each color channel of the image. Meanwhile, most natural images have a low-rank structure. Then we can fill in the missing elements of a partially observed image by the low-rank tensor completion method.

Figure 3. Test color images: airplane, baboon, barbara, boats, butterfly, house, lena, peppers (from left to right).

We use several standard images to evaluate our method. All the images, shown in Figure 3, have the same size of $256 \times 256 \times 3$. Table 1 presents the average PSNR values on this dataset with different sampling rates and the average runtime of all methods. Our method has achieved good performance under different sampling rates. SPC works well when the sampling rate is very low. In regard to algorithm runtime, our method is the fastest. It is about two times faster than TNN, which is the second fastest method. Figure 4 shows the inpainting results on two images under different removal masks. It is obvious that our method can also deal with the details of images. In summary, our method achieves good inpainting results and runs fast.

Table 1. Comparison of PSNR and runtime on the standard images.

Method	0.2	0.4	0.6	0.8	Runtime (s)
HaLRTC [15]	20.74	25.17	29.52	35.20	4.46
SPC [29]	**24.51**	27.58	30.45	34.15	39.83
T-SVD [31]	20.68	25.62	30.48	36.82	14.10
TNN [16]	22.02	26.65	31.22	36.92	2.19
TNN-DCT [17]	22.17	26.83	31.47	37.24	4.59
Ours	23.01	**28.20**	**33.13**	**39.09**	**0.91**

We also use the commonly used Berkeley Segmentation Database [1] (CBSD68) to evaluate the performance of methods. This database contains 68 color images, each of size $321 \times 481 \times 3$ or $481 \times 321 \times 3$. Table 2 shows the average PSNR values on CBSD68 with different sampling rates and the average runtime of all methods. It can be concluded that our method performs well and runs quite fast. These results confirm the advantages and robustness of our method.

Table 2. Comparison of PSNR and runtime on the CBSD68.

Method	0.2	0.4	0.6	0.8	Runtime (s)
HaLRTC [15]	22.30	27.40	31.77	37.72	13.17
SPC [29]	**26.37**	29.46	32.38	36.16	115.38
T-SVD [31]	23.67	29.08	35.37	44.37	39.37
TNN [16]	25.14	30.33	36.07	44.20	5.95
TNN-DCT [17]	25.32	30.57	36.43	44.51	10.82
Ours	26.04	**31.80**	**37.82**	**44.74**	**2.05**

4.2.2 Video Inpainting

In this experiment, we test our method and other methods on two YUV videos [2]. "Akiyo" and "Container" are relatively static and dynamic videos, respectively. Each video contains 300 frames, and the size of a frame is 144 × 176

[1] https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
[2] http://trace.eas.asu.edu/yuv/

(a) Orignal (b) Observation (c) HaLRTC (d) SPC (e) T-SVD (f) TNN (g) TNN-DCT (h) Ours

Figure 4. Results of image inpainting. (a) Original image. (b) Incomplete image. (c)-(h) Inpainting results of HaLRTC, SPC, T-SVD, TNN, TNN-DCT, Ours, respectively.

(a) Orignal (b) Observation (c) HaLRTC (d) SPC (e) T-SVD (f) TNN (g) TNN-DCT (h) Ours

Figure 5. Results of video inpainting. (a) Original frame of video. (b) Incomplete frame of video. (c)-(h) Inpainting results of HaLRTC, SPC, T-SVD, TNN, TNN-DCT, Ours, respectively.

pixels. For the sake of saving computation cost, we conduct numerical experiments by using the first 30 frames of the two videos and display the 15-th frame of the videos.

Table 3. Comparison of PSNR and runtime on the "Akiyo".

Method	0.2	0.4	0.6	0.8	Runtime (s)
HaLRTC [15]	26.09	30.88	35.46	41.24	8.20
SPC [29]	31.52	33.21	35.13	38.28	77.32
T-SVD [31]	35.25	40.46	44.83	49.88	43.80
TNN [16]	35.26	40.22	44.56	49.82	20.72
TNN-DCT [17]	36.40	42.43	47.58	53.80	26.44
Ours	**37.39**	**43.71**	**48.75**	**54.83**	**4.49**

Table 4. Comparison of PSNR and runtime on the "Container".

Method	0.2	0.4	0.6	0.8	Runtime (s)
HaLRTC [15]	23.95	28.26	32.57	38.08	7.88
SPC [29]	29.17	31.12	33.07	36.13	83.86
T-SVD [31]	33.52	38.50	42.48	47.71	42.98
TNN [16]	33.42	38.46	42.73	47.78	20.74
TNN-DCT [17]	36.10	44.35	49.69	54.57	27.02
Ours	**37.36**	**45.66**	**51.17**	**55.64**	**3.86**

Table 3 and Table 4 report the evaluation metrics of the two videos under different sampling rates, respectively. Figure 5 shows the inpainting results for the testing videos when the sampling rate is 0.2. Form the inpainting results and PSNR values, our method has achieved better recovery performance compared with other methods. As for the algorithm runtime, our method is still fastest. These results illustrate that the proposed method is effective and runs fast.

5. Conclusion

Benefited from the development of tensor theory, especially the t-product based on linear transform [8], we define a tensor average rank based on invertible real linear transform. We then propose a new tensor completion model using a nonconvex function to approximate the average rank. This surrogate can continuously approximate the tensor average rank. We develop an efficient algorithm for solving our tensor completion model and establish its convergence. Experimental results clearly illustrate that our method is competitive with other methods in terms of both recovery performance and efficiency.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0100602, in part by the National Natural Science Foundation of China under Grant 11771408 and 11871444.

References

[1] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[2] J.-F Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *Siam Journal on Optimization*, 20(4):1956–1982, 2008.

[3] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[4] C. J. Hillar and L.-H. Lim. Most tensor problems are np-hard. *Journal of the ACM*, 60(6), 2013.

[5] P. Jain, R. Meka, and I. S. Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945, 2010.

[6] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of ACM conference on Recommender systems*, pages 79–86, 2010.

[7] H. Kasai and B. Mishra. Low-rank tensor completion: a riemannian manifold preconditioning approach. In *Proceedings of International Conference on Machine Learning*, pages 1012–1021, 2016.

[8] E. Kernfeld, M. Kilmer, and S. Aeron. Tensor–tensor products with invertible linear transforms. *Linear Algebra and its Applications*, 485:545 – 570, 2015.

[9] H. A. L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105–122, 2000.

[10] M. E. Kilmer, K. S. Braman, N. Hao, and R. C. Hoover. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM Journal on Matrix Analysis and Applications*, 34(1):148–172, 2013.

[11] M. E. Kilmer and C. D. Martin. Factorization strategies for third-order tensors. *Linear Algebra and Its Applications*, 435(3):641–658, 2011.

[12] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *Siam Review*, 51(3):455–500, 2009.

[13] H. Kong, X. Xie, and Z. Lin. t-schatten-p norm for low-rank tensor recovery. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1405–1419, 2018.

[14] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[15] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013.

[16] C. Lu, J. Feng, Z. Lin, and S. Yan. Exact low tubal rank tensor recovery from gaussian measurements. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2504–2510, 2018.

[17] C. Lu, X. Peng, and Y. Wei. Low-rank tensor completion with a new tensor nuclear norm induced by invertible linear transforms. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5989–5997, 2019.

[18] N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics*, 37(1):246–270, 2009.

[19] N. Mesgarani, M. Slaney, and S. A Shamma. Discrimination of speech from nonspeech based on multiscale spectro-temporal modulations. *IEEE Transactions on Audio Speech and Language Processing*, 14(3):920–930, 2006.

[20] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *Siam Review*, 52(3):471–501, 2010.

[21] B. Romera-Paredes, H. Aung, N. Bianchi-Berthouze, and M. Pontil. Multilinear multitask learning. In *Proceedings of International Conference on Machine Learning*, pages 1444–1452, 2013.

[22] J. Salmi, A. Richter, and V. Koivunen. Sequential unfolding svd for tensors with applications in array signal processing. *IEEE Transactions on Signal Processing*, 57(12):4719–4733, 2009.

[23] L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[24] M. A. O. Vasilescu and D. Terzopoulos. Multilinear subspace analysis of image ensembles. In *Proceedings of European Conference on Computer Vision*, 2003.

[25] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2015.

[26] Q. Yao, J. T. Kwok, T. Wang, and T. Liu. Large-scale low-rank matrix learning with nonconvex regularizers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(11):2628–2643, 2019.

[27] T. Yokota, B. Erem, S. Guler, S. K. Warfield, and H. Hontani. Missing slice recovery for tensors using a low-rank model in embedded space. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 8251–8259, 2018.

[28] T. Yokota and H. Hontani. Simultaneous visual data completion and denoising based on tensor rank and total variation minimization and its primal-dual splitting algorithm. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3843–3851, 2017.

[29] T. Yokota, Q. Zhao, and A. Cichocki. Smooth parafac decomposition for tensor completion. *IEEE Transactions on Signal Processing*, 64(20):5423–5436, 2016.

[30] Y. Zhang and Z. Lu. Penalty decomposition methods for rank minimization. In *Advances in Neural Information Processing Systems*, pages 46–54. 2011.

[31] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer. Novel methods for multilinear data completion and de-noising based on tensor-svd. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3842–3849, 2014.

[32] P. Zhou, C. Lu, Z. Lin, and C. Zhang. Tensor factorization for low-rank tensor completion. *IEEE Transactions on Image Processing*, 27(3):1152–1163, 2018.